You are in an interaction window of the Edwin editor.
Type C-h for help.  C-h m will describe some commands.
Package: (user)

(cd "~/.cph/amord")
;Value 2: #[pathname 2 "/nfs/raid/home/gjs/cph/amord/"

(load "load")
;Loading "load.scm"
;Loading "../tms/load.scm"
;Loading "pgsql.com" -- done
;Loading "../snpob/load.scm"
;Loading "snpob-unx.pkd" -- done
;Loading "persistent.com" -- done
-- done
;Loading "tms-unx.pkd" -- done
;Loading "tms.com" -- done
;Loading "constraints.com" -- done
;Loading "numeric-constraints.com" -- done
;Loading "predicate-constraints.com" -- done
;Loading "tools.com" -- done
-- done
;Loading "representations.scm" -- done
;Loading "unify.scm" -- done
;Loading "compare.scm" -- done
;Loading "assertions-rules.scm" -- done
;Loading "interpreter.scm" -- done
;Loading "tools.scm" -- done
;Loading "logic.scm" -- done
-- done
;Value: #t
(make-amord-system 's)
;Value: #[system 4]

(trace-amord)
;Value: tracing

(full-logic s)
; Supported: (r1 (not (not ?a)) (subrule) ())
; Supported: (r2 (and ?a ?b) (subrule) ())
; Supported: (r3 (<-> ?a ?b) (subrule) ())
; Supported: (r4 (-> ?a ?b) (subrule) ())
; Supported: (r5 (not (or ?a ?b)) (subrule) ())
; Supported: (r6 (not (and ?a ?b)) (subrule) ())
; Supported: (r7 (and (not ?a) (not ?b)) (subrule) ())
; Supported: (r8 (or (not ?a) (not ?b)) (subrule) ())
; Supported: (r9 (show (not (not ?a))) (subrule) ())
; Supported: (r10 (show (and ?a ?b)) (subrule) ())
; Supported: (r11 (show (or ?a ?b)) (subrule) ())
; Supported: (r12 (show ?b) (subrule) ())
; Supported: (r13 (show (not ?a)) (subrule) ())
; Supported: (r14 (show (-> ?a ?b)) (subrule) ())
;Value: whew!

(amord
  '(assume (-> (human ?x) (fallible ?x)))
  s)
; Supported: (p1 (-> (human ?x) (fallible ?x)) (premise) (p1))
;Value: #[proposition 5]

(amord
  '(assume (-> (human ?x) (mortal ?x)))
  s)
; Supported: (p2 (-> (human ?x) (mortal ?x)) (premise) (p2))
;Value: #[proposition 6]

(amord
  '(assume (-> (greek ?x) (human ?x)))
  s)
; Supported: (p3 (-> (greek ?x) (human ?x)) (premise) (p3))
;Value: #[proposition 7]

(amord
  '(assume (show (-> (greek ?x) (mortal ?x))))
  s)
; Supported: (p4 (show (-> (greek ?x) (mortal ?x))) (premise) (p4))
;Value: #[proposition 8]
(deduce-until-done s)
; Supported: (r15 (-> (mortal ?x) (human ?x)) (subrule r4 p2) (p2))
; Supported: (r16 (-> (fallible ?x) (human ?x)) (subrule r4 p1) (p1))
; Supported: (r17 (-> (human ?x) (greek ?x)) (subrule p3 r4) (p3))
; Supported: (r18 (-> ?a (-> (greek ?x) (mortal ?x))) (subrule p4 r12) (p4))
; Supported: (r19 (or ?a (-> (greek ?x) (mortal ?x))) (subrule p4 r12) (p4))
; Supported: (r20 (or (-> (greek ?x) (mortal ?x)) ?a) (subrule p4 r12) (p4))
; Supported: (p5 (hypothesis (greek ?x)) (hypothesis) (p5))
; Supported: (r21 (hypothesis (greek ?x)) (subrule p4 r14) (p4))
; Supported: (p6 (greek ?x) (hi p5) (p5))
; Supported: (p7 (show (mortal ?x)) (bc:cp p5 p4) (p4 p5))
; Supported: (r22 (mortal ?x) (subrule p5 r21) (p4 p5))
; Supported: (r23 (-> ?a (mortal ?x)) (subrule r12 p7) (p4 p5))
; Supported: (r24 (or ?a (mortal ?x)) (subrule r12 p7) (p4 p5))
; Supported: (r25 (or (mortal ?x) ?a) (subrule r12 p7) (p4 p5))
; Supported: (p8 (show (human ?x)) (bc:mp p2 p7) (p4 p5 p2))
; Supported: (r26 (human ?x) (subrule p2 r23) (p4 p5 p2))
; Supported: (r27 (-> ?a (human ?x)) (subrule p8 r12) (p4 p5 p2))
; Supported: (r28 (or ?a (human ?x)) (subrule p8 r12) (p4 p5 p2))
; Supported: (r29 (or (human ?x) ?a) (subrule p8 r12) (p4 p5 p2))
; Supported: (p9 (show (greek ?x)) (bc:mp p8 p3) (p3 p4 p5 p2))
; Supported: (r30 (greek ?x) (subrule p3 r27) (p4 p5 p2 p3))
; Supported: (r31 (-> ?a (greek ?x)) (subrule p9 r12) (p3 p4 p5 p2))
; Supported: (r32 (or ?a (greek ?x)) (subrule p9 r12) (p3 p4 p5 p2))
; Supported: (r33 (or (greek ?x) ?a) (subrule p9 r12) (p3 p4 p5 p2))
; Supported: (p10 (human ?x) (mp p6 p3) (p3 p5))
; Supported: (p11 (mortal ?x) (mp p10 p2) (p2 p3 p5))
; Supported: (p12 (-> (greek ?x) (mortal ?x)) (cp p5 p11) (p2 p3 p5))
; Supported: (r34 (greek ?x) (subrule p12 r23) (p4 p2 p3 p5))
; Supported: (r35 (-> (mortal ?x) (greek ?x)) (subrule r4 p12) (p2 p3 p5))
;Value: done
(write-suppes-proof
 (the-proposition '(
   (-> (greek ?x) (mortal ?x)) s))

; p2   (-> (human ?x) (mortal ?x))                    (premise)  { p2 }
; p3   (-> (greek ?x) (human ?x))                     (premise)  { p3 }
; p6   (greek ?x)                                    (hi p5)    { p5 }
; p10  (human ?x)                                    (mp p6 p3) { p3 p5 }
; p11  (mortal ?x)                                   (mp p10 p2) { p2 p3 p5 }
; p5   (hypothesis (greek ?x))                       (hypothesis) { p5 }
; p12  (-> (greek ?x) (mortal ?x))                   (cp p5 p11) { p2 p3 }
; QED
#| Sandro’s encoding of a part of a law in N3
# 18usc228(a)(1) - second branch ($5K limit)
{  ?person s228:residesInDifferentStateFrom ?child.
    [ a s228:UnpaidObligation;
      s228:obligatedPerson ?person;
      s228:supportedPerson ?child;
      s228:amount ?amt
    ].
    ?amt math:greaterThan 5000.
}
=>
{  [ a law:ApparentViolation;
      law:alleged_violator ?person;
      law:law usc:18_228;
    ]
}
|#
(make-amord-system 's)

(trace-amord)

(full-logic s)

(amord
  '(rule (?c1 (unpaid (support-obligation ?p ?c)))
    (rule (?c2 (person ?p))
      (rule (?c3 (child ?c))
        (rule (?c4
          (not (equal (state-of-residence ?p)
                        (state-of-residence ?c))))
          (rule (?c5
            (amount-exceeds
             (support-obligation ?p ?c)
             5000dollars))
            (assert
             (Violation:18usc228 ?p)
             (18usc228 ?c1 ?c2 ?c3 ?c4 ?c5)))
          (rule (?c5
            (time-exceeds
             (support-obligation ?p ?c)
             1year))
            (assert
             (Violation:18usc228 ?p)
             (18usc228 ?c1
             ?c2
             ?c3
             ?c4
             ?c5))))))))

s)
(amord '(assume (-> (child ?x) (person ?x))) s)

(amord '(assume (person HarryDeadbeat)) s)

(amord '(assume (child JimDeadbeat)) s)

(amord '(assume
   (unpaid
     (support-obligation HarryDeadbeat JimDeadbeat)))
   s)

(amord '(assume
   (not (equal (state-of-residence HarryDeadbeat)
     (state-of-residence JimDeadbeat))))
   s)

(amord '(assume
   (amount-exceeds
     (support-obligation HarryDeadbeat JimDeadbeat) 5000dollars))
   s)

(amord '(assume (person JoeGoodfellow)) s)

(amord '(assume (child MaryGoodfellow)) s)

(amord '(assume
   (unpaid
     (support-obligation JoeGoodfellow MaryGoodfellow)))
   s)

(deduce-until-done s)
;;; Harry is a crook.

(write-suppes-proof
  (the-proposition '(Violation:18usc228 HarryDeadbeat) s))
; p4  (unpaid (support-obligation harrydeadbeat jimdeadbeat))
;      (premise)  { p4 }
; p5  (not (equal (state-of-residence harrydeadbeat)
;            (state-of-residence jimdeadbeat)))
;      (premise)  { p5 }
; p3  (child jimdeadbeat)  (premise)  { p3 }
; p6  (amount-exceeds (support-obligation harrydeadbeat
;                jimdeadbeat)
;      5000dollars)
;      (premise)  { p6 }
; p2  (person harrydeadbeat)  (premise)  { p2 }
; p10 (violation:18usc228 harrydeadbeat)
;     (18usc228 p2 p6 p3 p5 p4)  { p4 p5 p3 p6 p2 }
; QED

;;; But Joe is OK.

(write-suppes-proof
  (the-proposition '(Violation:18usc228 Joegoodfellow) s))
; Not supported!
;; With strategy also formalized.

(make-amord-system 's)

;(trace-amord)

(full-logic s)

(amd
 ' (rule (?goal (show (Violation:18usc228 ?p)))
   (assert (show (unpaid (support-obligation ?p ?c)))
   (18usc228:BC:0 ?goal))
   (rule (?cond1 (unpaid (support-obligation ?p ?c)))
   (assert (show (and (and (person ?p) (child ?c))
     (not (equal
       (state-of-residence ?p)
       (state-of-residence ?c))))))
   (18usc228:BC:1 ?goal ?cond1))
  (rule (?cond2
    (and (and (person ?p) (child ?c))
    (not (equal (state-of-residence ?p)
       (state-of-residence ?c))))))
  (assert (show
    (or (amount-exceeds
      (support-obligation ?p ?c) 5000dollars)
    (time-exceeds
      (support-obligation ?p ?c) 1year)))
   (18usc228:BC:2 ?goal ?cond1 ?cond2))
  (rule (?cond3
    (or (amount-exceeds
      (support-obligation ?p ?c) 5000dollars)
    (time-exceeds
      (support-obligation ?p ?c) 1year)))
   (assert (Violation:18usc228 ?p)
     (18usc228 ?cond1 ?cond2 ?cond3)))
))))

s)
(amord '(assume (-> (child ?x) (person ?x))) s)

(amord '(assume (person HarryDeadbeat)) s)

(amord '(assume (child JimDeadbeat)) s)

(amord '(assume
  (unpaid
    (support-obligation HarryDeadbeat JimDeadbeat)))
  s)

(amord '(assume
  (not (equal (state-of-residence HarryDeadbeat)
              (state-of-residence JimDeadbeat))))
  s)

(amord '(assume
  (amount-exceeds
    (support-obligation HarryDeadbeat JimDeadbeat)
    5000dollars))
  s)

(amord '(assume (person JoeGoodfellow)) s)

(amord '(assume (child MaryGoodfellow)) s)

(amord '(assume
  (unpaid
    (support-obligation JoeGoodfellow
                             MaryGoodfellow)))
  s)

(amord '(assume (show (Violation:18usc228 HarryDeadbeat)))
  s)

(deduce-until-done s)
;;; Harry is a crook.

(write-suppes-proof
 (the-proposition
   '(Violation:18usc228 HarryDeadbeat)
   s))

;; (unpaid (support-obligation harrydeadbeat jimdeadbeat))
(全省) { p4 }

;; (amount-exceeds (support-obligation harrydeadbeat
;;             jimdeadbeat) 5000dollars)
(全省) { p6 }

;; (or (amount-exceeds (support-obligation harrydeadbeat
;;                          jimdeadbeat) 5000dollars)
;;        (time-exceeds (support-obligation harrydeadbeat
;;                        jimdeadbeat) 1year))
(全省) { p23 }

;; (child jimdeadbeat) (premise) { p3 }

;; (person harrydeadbeat) (premise) { p2 }

;; (and (person harrydeadbeat) (child jimdeadbeat))
(全省) { p17 }

;; (not (equal (state-of-residence harrydeadbeat)
;;             (state-of-residence jimdeadbeat)))
(全省) { p5 }

;; (and (and (person harrydeadbeat) (child jimdeadbeat))
;;             (not (equal (state-of-residence harrydeadbeat)
;;                         (state-of-residence jimdeadbeat))))
(全省) { p19 }

;; (violation:18usc228 harrydeadbeat)
(全省) { p24 }

QED
;; Improvement to system: common subexpressions.

(amd
  'rule (?goal (show (Violation:18usc228 ?p)))
    (let ((cond1 (unpaid (support-obligation ?p ?c))))
      (assert (show :cond1) (18usc228:BC:0 ?goal))
      (rule ?cond1
        (let ((cond2
          (and (and (person ?p) (child ?c))
            (not (equal (state-of-residence ?p)
              (state-of-residence ?c))))))
          (assert (show :cond2) (18usc228:BC:1 ?goal ?cond1))
          (rule ?cond2
            (let ((cond3
              (or (amount-exceeds (support-obligation ?p ?c)
                5000dollars)
                (time-exceeds (support-obligation ?p ?c)
                  1year))))
              (assert (show :cond3)
                (18usc228:BC:2 ?goal ?cond1 ?cond2))
              (rule ?cond3
                (assert (Violation:18usc228 ?p)
                  (18usc228 ?cond1 ?cond2 ?cond3))
              )))))))))
)